Overzicht Assembler (alfabetisch)

Registers

AH	Deel EAX register (slide 3-11)
cr	controleregister
dr	dataregister
EAX	Accumulator
EBP	Basis Pointer
EBX	Basis
ECX	Counter
EDI	Destination index
EDX	Data
EFLAG	Flags
EIP	Instruction Pointer
ESI	Source index
ESP	Stack Pointer
init	controlebit; bit 2 van cr
ready	toestandsbit; bit 7 van tr
select	toestandsbit; bit 4 van tr
selectin	controlebit; bit3 van cr
strobe	controlebit; bit 0 van cr
tr	toestandsregister

Flags

a	Auxiliary Carry
c	**Carry**
d	Direction
i	Interrupt
io	I/O level
nt	Nested task
o	**Overflow**
p	Pariteit
s	**Sign**
t	Trap
z	**Zero**

Conditiecodes

a nbe	Above
ae nb	Above or equal
b nae	Below
be na	Below or equal
c	Carry
e	Equal
g nle	Greater
ge nl	Greater or equal
l nge	Less
le ng	Less or equal
nc	No carry
no	No overflow
np	No paritiy
ns	No sign
nz	No zero
o	Overflow
p	Parity
s	Sign
z	Zero

Adressen

mem8[x]	8 bit geven vanaf positie x
mem16[x]	16 bit geven vanaf positie x
mem32[x]	32 bit geven vanaf positie x

Instructies

adc doel, bron	Add with carry
add doel, bron	Tel bron op bij doel
addi doel, bron	Add immediate: tot 16 bits optellen bij een register
and d,s	d = d AND s
bal r, adres	Branch and link; deze instructie springt naar doel, en slaat het terugkeeradres op in r.
bsf d,s	Bit scan forward Deze instructie gaat op zoek naar het eerste niet-nul bit in de operand s en het resultaat komt in operand d Indien niets gevonden, wordt de z-flag aangezet en is het resultaat in d onbepaald
bsr d,s	Bit scan reverse (uitleg zie vorige)
bswap r	Wijzig de volgorde van de bytes in een multi-byte woord (wisselen tussen little endian en big endian)
bt d,o	Bit test
btc d, o	Bit test and complement
btr d, o	Bit test and reset
bts d,o	Bit test and set
call adres	Spring naar een subroutine
call r, adres	In plaats van het terugkeeradres op de stack op te slaan, kan het ook in een register worden opgeslagen
clc	toestandbit wijzigen. c = 0
cld	toestandbit wijzigen. d = 0
cli	toestandbit wijzigen. i = 0
cli	Onderbrekingen afzetten (onderbrekingsregelaar)
cmp d,s	d-s → vlaggen (vergelijkjen van waarden >, =, <) Doet hetzelfde als SUB, maar gooit het resultaat weg (met behoud van toestandsbits)
cmpxchg d,s	Compare and echange
dec al	Reg[al] = o ? 0ffh : 00h
dec doel	Doel = doel – 1
div bron	Deling (unsigned): eax = edx:eax / bron
enter n	Bewaart en definieert de stack frame poointer, en heeft dan nog de mogelijkheid om een aantal bytes als lokale veranderlijken te definiëren (door middel van de sub esp, n)
equ voorbeeld: dr eque 378h	Wordt gebruikt om een symbolische constante te definiëren, te vergelijken met C's define. Dus het voorbeeld: dr wordt een symbolische constante 378 (hexadecimal)
f2xm1	FP exponent st = 2st-1
fabs	FP absolute waarde
fadd	Flowting Point Add St(1) = st+st(1); pop
fadd	Flowting Point telling
fadd <ae>	St = st+mem[ae]
fadd st(i)	St = st + st(i)
fadd st(i), st	St(i) = st+st(i)
fadd st, st(i)	St = st(i)+st

faddp st(i), st	St(i) = st+st(i); pop
fbld	Push BCD-getal (ld = load)
fbstp	Store BCD and pop
fchs	FP negatie
fcomi	FP compare and set eflags
fcos	FP cosinus
fdiv	FP deling
fild	Push integer 16/32/64 bit
fist	Store integer 16/32/64 bit
fld	Push floating point value 32/64/80 bit
fld1	Push 1.0
fldl2	Push $log_e(2)$
fldl2e	Push $log_2(e)$
fldl2t	Push $log_2(10)$
fldlg2	Push $log_{10}(2)$
fldpi	Push getal Pi
fldz	Push +0.0
fmul	FP vermenigvuldiging
fpatan	FP arcus tangens
fprem	FP rest na deling
fptan	FP tangens
fsin	FP sinus
fsqrt	FP vierkantswortel
fst	Store floating point value 32/64/80 bit
fsub	FP aftrekking
fyl2x	FP logaritme: st(1) = st(1)*log2(st);pop
fyl2xp1	FP logaritme: st(1) = st(1) * log2(st+1); pop
idev bron	Deling (signed): edx = edx:eax % bron
imul bron imul d, bron imul d, bron1, bron2	Vermenigvuldiging (signed): exd:eax=bron * eax d = d * bron d = bron1 * bron2
in register, port	Stuur inhoud van “register” naar “poort”
inc doel	Doel = doel + 1
ja	Jump if above
jae	Jump if above or equal
jb	Jump if below
jbe	Jump if below or equal
jc	Jump if carry
je	Jump if equal
jg	Jump if greater
jge	Jump if greater or equal
jl	Jump if less
jle	Jump if less or equal
jmp adres	Spring naar adres (reg[eip] = adres)
jmp r	Ga terug naar adres in r
jnc	Jump if not carry
jno	Jump if not overflow
jnp	Jump if not parity
jns	Jump if not sign

jnz	Jump if not zero
jo	Jump if overflow
jp	Jump if parity
js	Jump if sign
jz	Jump if zero
lahf	Plaats de status in register ah
ld register, bron	Laad de inhoud van “bron” in register
leave	De leave instructie herstelt de esp zodanig dat hij naar de vorige stack frame pointer wijst, en haalt de vorige stack frame pointer dan op. Dan is de functie in principe klaar om de return-instructie uit te voeren.
lock	Een zogenaamde prefix. De semantiek van de instructie die eorop volgt wordt licht gewijzigd. Het lock-prefix wordt gebruikt om de instructie die erop volgt atomair uit te voeren. Kan gebruikt worden met instructies: add, adc, and, btc, btr, bts, cmpxchg, dec, inc, neg, not, or, sbb, sub, xor, xadd, xchg; en dat op voorwaarde dat deze instructies inwerken op een geheugenoperand
lodsb	Load string byte: register al krijgt de inhoud van wat staat op het geheugenadres dat staat in register esi (reg[al] = reg[esi]; reg[esi] += 1)
lodsd	Load string dword: register eax krijgt de inhoud van wat staat op het geheugenadres dat staat in register esi (reg[eax] = reg[esi]; reg[esi] += 1)
lodsw	Load string word: register ax krijgt de inhoud van wat staat op het geheugenadres dat staat in register esi (reg[ax] = reg[esi]; reg[esi] += 1)
loop	Deze instructie decrementeerd ecx en springt nadien naar het opgeven adres indien ecx verschilt van 0.
lui doel, bron	Load upper immediate: laad de hoogste bits in register doel en zet de laatste op 0
lw doel, bron	Load 32 word
mov doel, bron mov eax, ebx mov ebx, [4] mov [ebp], ebx mov byte doel, bron mov eax, 0aah	Kopieer inhoud van “bron” naar “doel” Kopieert de inhoud van register ebx naar eax Kopieer inhoud van geheugenlocatie 4 naar ebx Kopieer de inhoud van ebx naar de plaats die staat in het register ebp Hierbij opgeven dat de lengte 1 byte is. Ook mogelijk voor word en dword. Plaats constante 0aah in register eax (voorafgegaan door 0 = constante)
movsb	Move string byte: de inhoud die staat op het geheugenadres dat staat in register edi is de inhoud van esi. (reg[edi] = reg[esi]; reg[esi] += 1)
movsd	Move string dword: de inhoud die staat op het geheugenadres dat staat in register edi is de inhoud van esi. (reg[edi] = reg[esi]; reg[esi] += 1)
movsw	Move string word: de inhoud die staat op het geheugenadres dat staat in register edi is de inhoud van esi. (reg[edi] = reg[esi]; reg[esi] += 1)
mul bron	Vermenigvuldiging (unsigned): edx:eax = bron * eax
neg doel	Teken omkeren (doel = - doel)
nop	No operation: doe niets
not d	d = NOT d
or d,s	d = d OR s
out port, register	Haal de inhoud van “poort” en kopieer deze naar “register”
pop doel	Verwijder het bovenste element van de stack en plaats de inhoud in “doel”
popad	Pop van edi, esi, ebp, -, ebx, edx, ecx, eax
popfd	Toestandsregister van de stack halen
push doel	Voeg de inhoud van “doel” op de stack

Pushl variabele	Push de “variabele” op de esp stack
pushad	Push van eax, ecx, edx, ebx, (oude esp), ebp, esi, edi
pushfd	Toestandsregister plaatsen op stack
rcl d,n	uitgebreide bitrotatie naar links
rcr d,n	uitgebreide bitrotatie naar rechts
rep movsb	Voer movsb uit totdat het register ecx op 0 staat
ret	Ga terug naar de routine die je heeft opgeroepen
rol d,n	bitrotatie naar links (weggeschoven bits worden er langs de andere kant bijgevoegd)
ror d,n	bitrotatie naar rechts
sahf	Kopieer de inhoud van ah naar status (toestandsregister)
sal d,n	Shift arithmetic left: aritmetische d = d << n exact gelijk aan SHL operatie
sar d,n	Shift arithmetic right: aritmetische d = d >> n; ingevoerde bits dezelfde waarde als de tekenbit van de verschoven waarde
sbb doel, bron	Substract with borry
set <cc> d	D = conditiecode Deze instructie kopieert een particuliere conditiecode naar de operand d die op basis hiervan de waarde 1 of 0 krijgt.
setge al	Reg[al] = ge ? 1 : 0
setno al	
seto ah	Reg[ah] = o ? 1 : 0
shl d,n	Shift Left: d = d << n (ingevoerde bits zijn 0)
shld d,s,n	d = (d:s << n) <63:32>
shr d,n	Shft Right: d = d >> n
shrd d,s,n	d = (d:s >> n) <63:32>
slodsb	Store string dword (4 byte): de inhoud van het geheugenadres dat in register edi krijgt de inhoud van register eax (reg[edi] = reg[eax]; reg[edi] += 1)
st register, doel	Bewaar de inhoud van “register” in “doel”
stc	toestandbit wijzigen. c = 1
std	toestandbit wijzigen. d = 1
sti	toestandbit wijzigen. i = 1
sti	Onderbrekingen terug aanzetten (onderbrekingsregelaar)
stosb	Store string byte: de inhoud van het geheugenadres dat in register edi krijgt de inhoud van register al (reg[edi] = reg[al]; reg[edi] += 1)
stosw	Store string word (2 byte): de inhoud van het geheugenadres dat in register edi krijgt de inhoud van register ax (reg[edi] = reg[ax]; reg[edi] += 1)
sub doel, bron	Aftrekking van bron bij doel
test d,s	d AND s → vlaggen (testen van bits: aan, uit) Doet altijd hetzelfde als AND, maar gooit het resultaat weg, met behoud van de toestandsbits.
xadd	Exchange and add: vervang s door d en overschrijf d met de som van de oorspronkelijke d en s. t = d+s; s = d, d = t
xchg doel, bron	Wissel de inhoud van 2 locaties
xor d,s	d = d XOR s